

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-021031

(43)Date of publication of application : 24.01.1995

51)Int.Cl.

G06F 9/45

21)Application number : 05-163384

(71)Applicant : NEC CORP

22)Date of filing : 01.07.1993

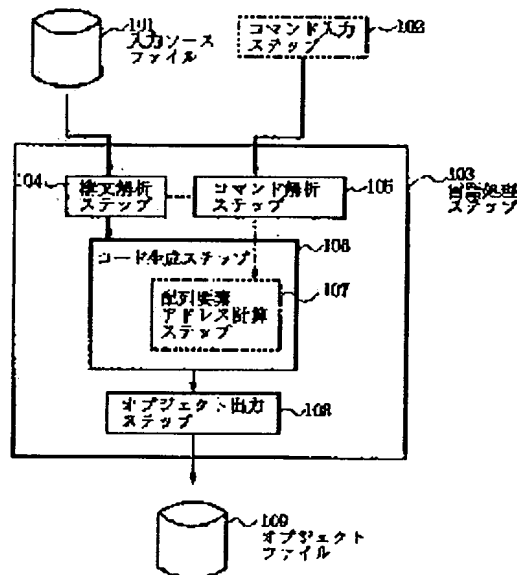
(72)Inventor : ISOZAKI HIROKO

## 54) LANGUAGE PROCESSING METHOD

### 57)Abstract:

**PURPOSE:** To accelerate the execution of an object to be generated, and to shorten the size by analyzing command designation and selecting any optimum calculating method concerning the address calculating method of arranged elements.

**CONSTITUTION:** In a command analysis step 105, the command designation to designate the size of arrangement is analyzed, when the reference of arranged elements is recognized in a syntax analysis step 104, on the other hand, in an arranged element address calculation step 107, calculation for calculating offset from the head of arrangement is performed by selecting any suitable data width based on command information analyzed in advance, and an efficient arranged element address calculation object is generated in an object output step.



## LEGAL STATUS

Date of request for examination]	01.07.1993
Date of sending the examiner's decision of rejection]	05.01.1999
Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
Date of final disposal for application]	
Patent number]	3233245
Date of registration]	21.09.2001
Number of appeal against examiner's decision of rejection]	11-01732
Date of requesting appeal against examiner's decision of rejection]	04.02.1999
Date of extinction of right]	21.09.2004

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-21031

(43) 公開日 平成7年(1995)1月24日

(51) Int.Cl.<sup>6</sup>

識別記号

庁内整理番号

F I

技術表示箇所

G 0 6 F 9/45

9292-5B

G 0 6 F 9/44

3 2 2 H

審査請求 有 請求項の数 1 O L (全 8 頁)

(21) 出願番号 特願平5-163384

(22) 出願日 平成5年(1993)7月1日

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 磯崎 博子

東京都港区芝五丁目7番1号 日本電気株式会社内

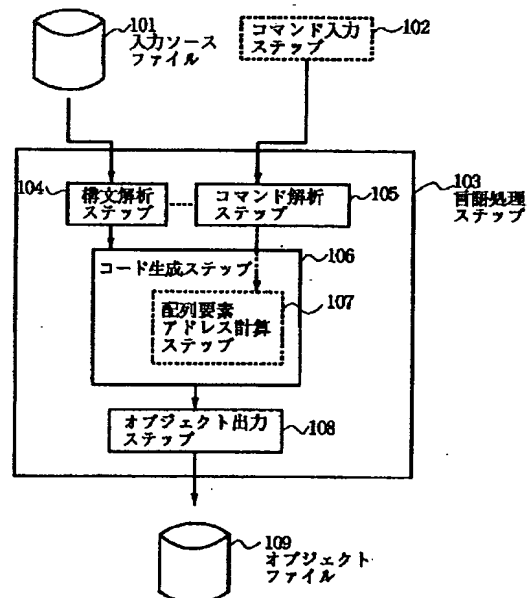
(74) 代理人 弁理士 京本 直樹 (外2名)

(54) 【発明の名称】 言語処理方法

(57) 【要約】

【目的】 配列要素のアドレス計算方法において、コマンド指定を解析し最適な計算方法を選択することにより、生成するオブジェクトの実行時間の向上とサイズの短縮を図る。

【構成】 コマンド解析ステップ105で配列の大きさを指定したコマンド指定を解析する一方、配列要素の参照を構文解析ステップ104で認識したならば、配列要素アドレス計算ステップ107において先に解析したコマンド情報に基づき配列の先頭からのオフセットを求める計算を適切なデータ幅を選択して行ない、効率の良い配列要素アドレス計算オブジェクトをオブジェクト出力ステップで生成する。



## 【特許請求の範囲】

【請求項1】 ソースプログラム情報を入力するステップと、前記ソースプログラム情報を解析するステップと、オブジェクト情報の生成時に処理する配列のアドレス計算をするステップと、前記オブジェクト情報を出力するステップとを有する言語処理プログラムにおいて、前記配列のアドレス計算をするステップが配列の最大サイズを指定したコマンド入力解析して得たコマンド情報から最短のデータ幅を選択するステップと、前記選択したデータ幅で配列の先頭からのオフセットを求める計算を行なうステップとを有することを特徴とする言語処理方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明は、言語処理方法に関し、特に配列構造を有する言語の言語処理方法に関する。

## 【0002】

【従来の技術】言語処理プログラムが入力したソースプログラムを翻訳し、その結果として生成したファイルをオブジェクトモジュール（以下、オブジェクトと呼ぶ）と言う。一般に、言語処理プログラムによって処理されるソフトウェアはその実行性能として速さが求められる。これを実現するためには言語処理を行なった結果であるオブジェクトのサイズが小さいこと、およびオブジェクトの実行が早いことが必要である。一方、ソフトウェアの規模の増大に伴い、ハードウェアのデータ空間は拡大する方向にある。

【0003】従来の言語処理方法では、このデータ空間の拡大に伴い、配列要素のデータ空間中の位置を計算するアドレス計算に関しても、データ空間の大きさに合わせて行なっている。仮に、データ空間を4ギガバイト（＝ $2^{32}$ バイト）だとすると、このデータ空間のアドレスを表現するには32ビット＝4バイト必要である。この仮定に基づき、図9を参照して従来の配列要素のアドレス計算の方法について説明する。

【0004】配列要素の先頭からのオフセットを計算する処理では、まず配列の添字を取り出す（ステップ901）。次に添字の内容を4バイトに拡張する（ステップ902）。そして、拡張後の添字の内容（4バイト）と配列要素のデータ幅を乗算してオフセットを求める（ステップ903）。以上がオフセットの計算の処理である。

【0005】次に、配列要素のアドレスは、配列の先頭アドレスを取り出し（ステップ904）、取り出した配列先頭アドレスと先に求めたオフセットを加算する（ステップ905）ことにより求める。

【0006】言語プログラムは、以上の計算方法に対応したオブジェクトを生成していた。ステップ902で添字の内容を4バイトに拡張しているのは次の理由による。一般に高級言語では複数モジュールのリンクを可能

とするために、変数の外部参照を許しているが、配列も同様に外部参照した場合、配列の大きさが不明となる。そのため、求めるオフセットの最大値がデータ空間と同じ大きさを持つ可能性を考慮して4バイトに拡張する処理を行なっている。

## 【0007】

【発明が解決しようとする課題】上述した従来の方式では、オフセットを求めるための乗算の前に添字を4バイトに拡張してしまうために、乗算の一方の項が4バイトとなり乗算を行なう負荷が大きくなる問題が発生する。これは、4バイトデータをオペランドとして指定可能な乗算命令を持つアーキテクチャをターゲットとしている言語処理では問題はないが、16ビット＝2バイト、あるいはそれ以下のビット幅を中心とした命令を備えるアーキテクチャをターゲットとする場合には、2バイトデータを取り扱うのと4バイトデータを取り扱う方とでは後者の方が命令サイズが長い。つまり、データ空間のアドレス幅よりもアーキテクチャのデータ幅が小さい場合に、前述の前列要素のアドレス計算方法をとっていると生成するオブジェクトのサイズが大きく、かつオブジェクトの実行性能が悪くなってしまう。

【0008】このような問題を解決する方法として、16ビットアーキテクチャマシンで一時的に取り扱うデータ空間が、16ビットでアドレスを表現できる大きさであった。このために配列の計算においても展示を16ビット＝2バイトに拡張すれば良く、2バイトデータともう1つのデータとの乗算でオフセットを求めることになるので、もう1つのデータが2バイトよりも小さければ一命令で乗算を行なうことが可能であった。

【0009】しかしながら、最近ではアーキテクチャのデータ幅以上に大きなデータ空間が求められる傾向にあり、配列要素のアドレス計算においても、前述の通りオブジェクトサイズが大きく実行性能も悪いオブジェクトを生成してしまう問題が生じている。

【0010】一方、データ空間が拡大されても、多数のデータを取り扱うだけで、必ずしも1つの配列の大きさが4バイトの大きさである必要はない。

【0011】したがって、本発明の目的は、アーキテクチャのデータ幅以上に大きなデータ空間を持つハードウェアをターゲットとする言語処理において、配列要素のアドレス計算を効率良く行い、オブジェクトをより小さくするとともに、オブジェクトの実行性能を向上することにある。

## 【0012】

【課題を解決するための手段】本発明の言語処理方法は、ソースプログラム情報を入力するステップと、前記ソースプログラム情報解析するステップと、オブジェクト情報の生成時に処理する配列のアドレス計算をするステップと、前記オブジェクト情報を出力するステップとを有する言語処理プログラムにおいて、前記配列のアド

3

4

レス計算をするステップが配列の最大サイズを指定したコマンド入力を解析して得たコマンド情報から最短のデータ幅を選択するステップと、前記選択したデータ幅で配列の先頭からのオフセットを求める計算を行なうステップとを有している。

【0013】

【実施例】次に、図面を参照して本発明の第1の実施例の言語処理方法を詳細に説明する。図1は本実施例のシステムブロック図である。

【0014】言語処理ステップ103は、入力ソースファイル101を入力して解析を行なう構文解析ステップ104と、コマンド入力102を入力して解析するコマンド解析ステップ105と、構文解析された内容に従ってコード生成を行なうコード生成ステップ106と、生成されたコードをオブジェクトファイル109に出力するオブジェクト出力ステップ108とから構成される。さらに、コード生成ステップ106の中にはコマンド解析ステップ105の解析結果に従って適切な計算方法を選択する配列要素アドレス計算ステップ107を含む。

【0015】本実施例では、コマンド入力102で入力ソースファイル中に現われる配列の最大の大きさがターゲットマシンのint幅を超えるか否かを指定する。ここではintを2バイトとして、配列の最大の大きさが64Kバイトを超えるか否かを指定することになる。コマンド入力102の例を図2に示す。“\_short\_array”がこの例のコマンド指定である。コマンド解析部105ではこの指定を解析して、計算方法指定データとして保存する。

【0016】構文解析ステップ104で配列要素の参照を認識した場合は、まず配列要素のアドレスを計算しなければならないので、コード生成ステップ106の配列要素アドレス計算ステップ107を呼び出す。配列要素アドレス計算ステップ107ではコマンド解析ステップ105の解析結果で得られた計算方法指定データに従って、2通りの配列要素アドレス計算アルゴリズムのいずれかを選択する。

【0017】配列要素アドレス計算107の手順を図3を参照して以下に説明する。

【0018】まず、計算方法指定データが配列がintの大きさに収まることを示しているかを調べる(ステップ301)。intの大きさに収まるならば生成するオブジェクトとして配列要素アドレス計算アルゴリズム304を選択してオブジェクト生成を行なう(ステップ302)。intの大きさに収まらないならば従来と同一の配列要素アドレス計算アルゴリズム305を選択してオブジェクト生成を行なう(ステップ303)。配列要素アドレス計算アルゴリズム304は、配列の添字を取り出す(ステップ306)。

【0019】次に、取り出した添字の内容(2バイト)と配列要素のデータ幅の乗算を行なってオフセットを先

に求める(ステップ307)。ここで配列要素のデータ幅が2バイトを超えるならば複数回に分けて乗算を行なう必要があるが、添字側は2バイトなので4バイト×4バイトの乗算よりも負荷は少ない。オフセットを求めたならばそれをアドレス幅(4バイト)に拡張する(ステップ308)。これは次にアドレスとの加算を行なうのでデータ幅を合わせる必要があるからである。以上で配列の先頭からのオフセットが求められる。

【0020】次に、配列の先頭アドレスを取り出し(ステップ309)、取り出した配列の先頭アドレスとステップ308で求めたオフセットを加算して配列要素のアドレスを求める(ステップ310)。配列計算アルゴリズム305は、従来技術と同一の配列要素アドレス計算アルゴリズムである。

【0021】配列要素参照の例を図4に示す。

【0022】配列tabは外部参照をしているので大きさが不明である。そして、“tab[i]”についてアドレス計算が必要になっている。これを前述の手順に基づいてオブジェクトを生成した結果を図5および図6に示す。図5は配列の大きさがint以下であることをコマンドで指定した場合、図6は配列の大きさがintを超えることを指定した場合を示す。

【0023】本実施例では、配列要素のデータ幅が4バイトであるので、乗算部分で乗算命令を使用せずにシフト命令を用いている。図5および図6に示す通り、配列の大きさがint以下であることを指定した場合はオブジェクトサイズが合計19バイト、配列の大きさがintを超える場合は合計20バイトとなる。図6に示された生成コードは従来出力していたものと同じであるから、本実施例において配列の大きさがint以下であることを指定した場合は、従来のオブジェクトサイズよりも1バイト減少することがわかる。

【0024】次に、本発明の第2の実施例の言語処理方法について説明する。

【0025】上記第1の実施例は、コマンド入力によって配列の大きさがintを超えるか否かを指定させたが、全ての配列がint以下/intより大きいのではなく配列毎にintを超えるか否かが異なる場合がある。この場合は、入力ソースファイル中にコマンドを埋め込む手段によって最適な配列要素アドレス計算アルゴリズムを選択することが可能である。

【0026】図7を参照すると言語処理ステップ702は、構文解析ステップ703の中にコマンド解析ステップ704が含まれること、入力ソースファイル701以外への入力データが存在しないこと以外は、第1の実施例のステップと同じステップを有している。

【0027】本実施例では図8に示す通り、配列の大きさがintを超えるか否かを入力ソースファイル701の中に記述する。図8に示す配列tab1は配列の大きさがint以下であることを示すコマンド”short

-array記号が宣言部に記述されている。

【0028】構文解析ステップ03が配列の宣言を認識したならば、コマンド解析ステップ704でintを超えるか否かのコマンド指定の有無から、その配列の大きさがintを超えるか否かを判断する。そして、コマンド解析ステップ704の判断に基づき、配列要素の参照を認識した場合は、コード生成ステップ705の配列要素アドレス計算ステップ706を呼び出す。以降の処理については第1の実施例と同じ処理であるので詳細な説明は省略する。

【0029】この第2の実施例は、配列のそれぞれに対して配列の大きさを指定できるので、第1の実施例よりもきめ細かなアドレス計算方法の選択が可能である。

【0030】

【発明の効果】以上、説明したように本発明によれば、コマンドによって指定された配列の大きさに対応して最適な配列要素のアドレス計算アルゴリズムを選択することにより、生成するオブジェクトのサイズを短縮することが可能である。

【0031】配列の大きさがint以下である場合を例にとると、第1の実施例で説明した通り、コマンド指定により1回の配列要素のアドレス計算につき1バイト短縮することが可能である。第1の実施例のデータはC言語のlongデータ（4バイト）のシフト演算が存在するマシンアーキテクチャをターゲットとしたものだが、longデータのシフト演算が存在しないマシンの場合は従来技術の計算方法ではlongデータのシフトをライブラリコールによって実現することになり、増加するオブジェクトサイズ、増加した命令の実行時間分、本発明による計算方法の法が有利となる。

【0032】すなわち、生成するオブジェクトのサイズを短縮することにより、プログラム空間を節約することができるとともに、オブジェクトのロード時間の短縮を

図ることも可能となる。

【図面の簡単な説明】

【図1】本発明の第1の実施例の言語処理方法全体の構成図である。

【図2】図1に示す言語処理方法のコマンド入力例を示す図である。

【図3】図1に示す言語処理方法の配列要素アドレス計算アルゴリズム選択処理の流れ図である。

【図4】入力ソースファイルの例（第1の実施例）を示す図である。

【図5】配列要素アドレス計算アルゴリズム304による出力オブジェクト例を示す図である。

【図6】配列要素アドレス計算アルゴリズム305および従来技術による出力オブジェクト例を示す図である。

【図7】本発明の第2の実施例の言語処理方法全体の構成図である。

【図8】図7に示す言語処理方法の入力ソースファイル例を示す図である。

【図9】従来の言語処理方法の配列要素アドレス計算アルゴリズムの流れ図である。

【符号の説明】

101, 701 入力ソースファイル  
102 コマンド入力ステップ  
103, 702 言語処理ステップ  
104, 703 構文解析ステップ  
105, 704 コマンド解析ステップ  
106, 705 コード生成ステップ  
107, 706 配列要素アドレス計算ステップ  
108, 707 オブジェクト出力部ステップ  
109, 708 オブジェクトファイル  
301~315 配列要素アドレス計算アルゴリズム  
選択処理  
901~905 従来の配列要素アドレス計算処理

【図2】

```
ccxx -short_array ... 入力ソースファイル名
```

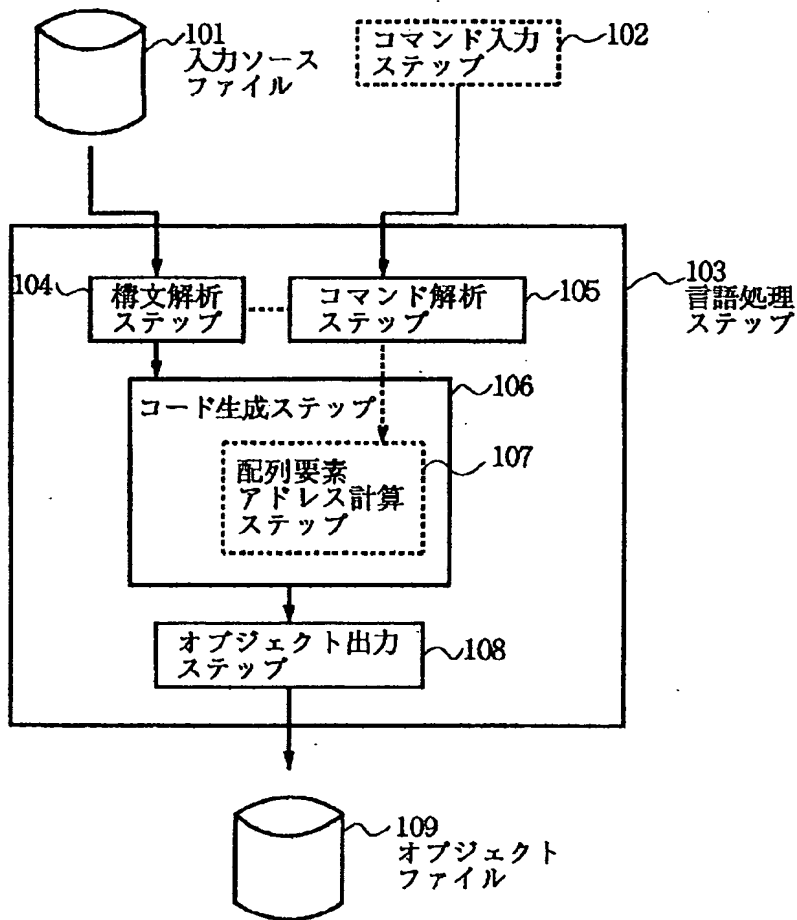
【図5】

```
movw  r0,B:8[rp5]  ;i  (4バイト)
shlw   r0,04        (2バイト)
cnvwld rp0          (2バイト)
add    rp0,#_tab     (6バイト)
movw   [rp0],#1      ;1  (5バイト)
```

【図4】

```
extern long tab[];
void func(int i)
{
    tab[i]=1;
}
```

【図1】



【図6】

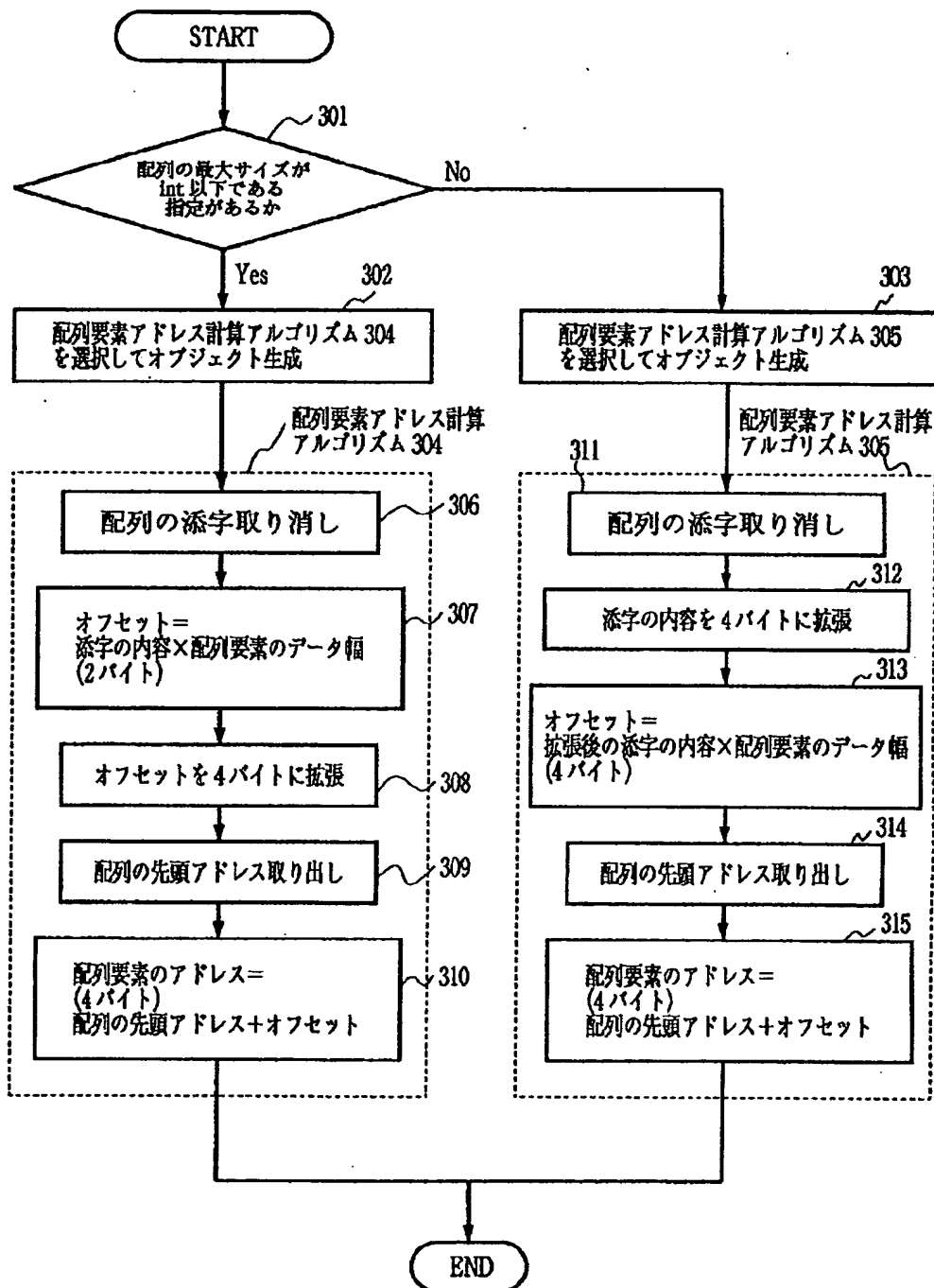
movw	r0,B:8[rp5]	; i	(4 バイト)
cnvwd	rp0		(2 バイト)
shld	rp0,04		(3 バイト)
add	rp0,#_tab		(6 バイト)
movw	[rp0],#1	; i	(5 バイト)

【図8】

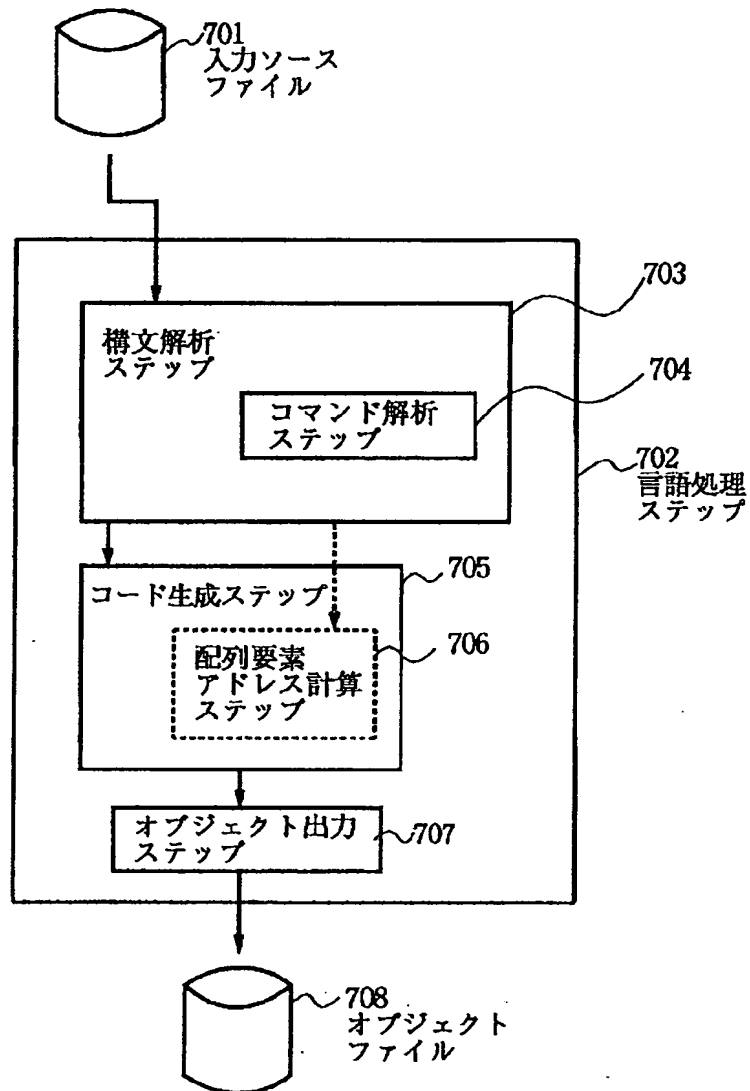
```

short_array extern long tab1[];
extern int tab2[];
void func (int i, int j)
{
    tab1[i] = 1;
    tab[j] = 1;
}
  
```

【図3】



【図7】





【図9】

